Distributed Control of the Laplacian Spectral Moments of a Network

Victor M. Preciado, Michael M. Zavlanos, Ali Jadbabaie, and George J. Pappas

***Abstract*— It is well-known that the eigenvalue spectrum of the Laplacian matrix of a network contains valuable information about the network structure and the behavior of many dynamical processes run on it. In this paper, we propose a *fully decentralized* algorithm that iteratively modifies the structure of a network of agents in order to control the moments of the Laplacian eigenvalue spectrum. Although the individual agents have knowledge of their *local* network structure only (i.e., myopic information), they are collectively able to aggregate this local information and decide on what links are most beneficial to be added or removed at each time step. Our approach relies on gossip algorithms to distributively compute the spectral moments of the Laplacian matrix, as well as ensure network connectivity in the presence of link deletions. We illustrate our approach in nontrivial computer simulations and show that a good final approximation of the spectral moments of the target Laplacian matrix is achieved for many cases of interest.**

I. Introduction

A wide variety of distributed systems composed by autonomous agents are able to display a remarkable level of self-organization despite the absence of a centralized coordinator [1, 2]. For example, the structure of many "self-engineered" networks, such as social and economic networks, emerges from local interactions between agents aiming to optimize their local utilities [3]. Motivated by the implications of a network's Laplacian spectrum on its structure (i.e., number of connected components) and behavior of dynamical processes implemented on it (i.e., speed of convergence of distributed consensus algorithms), we propose a distributed model of graph evolution in which autonomous agents can modify their local neighborhood in order to control a set of moments of the network Laplacian spectrum.

The eigenvalue spectra of a network provide valuable information regarding the behavior of many dynamical processes running within the network [4]. For example, the eigenvalue spectrum of the Laplacian matrix of a graph affects the mixing speed of Markov chains [5], the stability of synchronization of a network of nonlinear oscillators [6, 7], the spreading of a virus in a network [8, 9], as well as the dynamical behavior of many decentralized network algorithms [10]. Similarly, the second smallest eigenvalue of the Laplacian matrix (also called spectral gap) is broadly considered a critical parameter that influences the stability and robustness properties of dynamical systems that are implemented over information networks [11, 12]. Optimization of the spectral gap has been studied by several authors both in a centralized [13]–[15] and decentralized context [16]. In contrast, our approach focuses on controlling the moments of the Laplacian eigenvalue spectrum. In this way, we can influence the behavior of certain dynamical processes run within the network. As we show, the benefit of controlling the spectral moments, and especially the lower order ones, lies in lower computational cost and elegant distributed implementation.

A major challenge in our approach is to efficiently control the spectral moments of a network in a *fully distributed fashion* while maintaining *network connectivity* in the presence of link deletions. Our work is related to [17], where a fully distributed algorithm is proposed to compute the full set of eigenvalues and eigenvectors of a matrix representing the network topology. However, our approach is computationally cheaper since computation of the spectral moments does not require a complete eigenvalue decomposition, but can be performed distributively by averaging local network information, such as node degrees. On the other hand, control of the network structure to the desired set of spectral moments is based on greedy actions (link additions and deletions) that are the result of distributed agreement protocols between the agents. We show that our distributed topology control algorithm is stable and converges to a network with spectral moments "close" to the desired. The performance of our algorithm is illustrated in computer simulations.

The rest of this paper is organized as follows. In Section II, we formulate the problem under consideration and review some terminology. In Section III, we derive closed-form expressions for the first four moments of the Laplacian spectrum in terms of graph properties that can be measured locally. Based on these expressions, we introduce a distributed algorithm to compute these moments. In Section IV, we propose an efficient distributed algorithm to control of the spectral moments of a network. Finally, in Section V, we illustrate our approach with several computer simulations.

II. Preliminaries & Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph on n nodes, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. If $(v_i, v_j) \in \mathcal{E}$ whenever $(v_j, v_i) \in \mathcal{E}$ we say that $\mathcal{G}$ is *undirected* and call nodes v_i and v_j *adjacent* (or neighbors), which we denote by $v_i \sim v_j$. The set of all nodes adjacent to node v constitutes the *neighborhood* of node v, defined by $\mathcal{N}^v = \{w \in \mathcal{V} \ : \ (v, w) \in \mathcal{E}\}$, and the number of those neighbors is called the *degree* of node

This work is partially supported by the ONR MURI HUNT and AFOR MURI Complex Networks programs

Victor M. Preciado, Michael M. Zavlanos, Ali Jadbabaie, and George J. Pappas are with GRASP Laboratory, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA 19104, USA {preciado,zavlanos,jadbabai,pappasg}@seas.upenn.edu

v, denoted by $\deg v$. In this paper, we consider finite *simple* graphs, meaning that two nodes are connected by at most one edge and self-loops are not allowed.

We define a *walk* from v_0 to v_k of length k to be an ordered sequence of nodes $(v_0, v_1, ..., v_k)$ such that $v_i \sim v_{i+1}$ for $i = 0, 1, ..., k-1$. We say that a graph $\mathcal{G}$ is *connected* if there exists a walk between every pair of nodes. If $v_0 = v_k$, then the walk is closed. A closed walk with no repeated nodes (with the exception of the first and last nodes) is called a *cycle*. *Triangles* and *quadrangles* are cycles of length three and four, respectively. Let $d(v, w)$ denote the *distance* between two nodes v and w, i.e., the minimum length of a walk from v to w. We say that v and w are k-th order neighbors if $d(v, w) = k$ and define the k-th order neighborhood of a node v as the set of nodes within a distance k from v, i.e., $\mathcal{N}_k^v = \{w \in \mathcal{V} : d(v, w) \leq k\}$. A k-th order neighborhood, induces a subgraph $\mathcal{G}_k^v \subseteq \mathcal{G}$ with node-set $\mathcal{N}_k^v$ and edge-set $\mathcal{E}_k^v$ defined as the set of edges in $\mathcal{E}$ that connect two nodes in $\mathcal{N}_k^v$. We say that a graphical property $P_{\mathcal{G}}$ is *locally measurable within a radius k around a node v* if $P_{\mathcal{G}}$ is exclusively a function of the neighborhood subgraph, i.e., $P_{\mathcal{G}} = f(\mathcal{G}_k^v)$. For example, both the degree and the number of triangles touching a node are locally measurable within a radius 1. Also, the number of quadrangles touching a node is locally measurable within a radius 2.

Graphs can be algebraically represented via the *adjacency* and *Laplacian* matrices. The *adjacency matrix* of an undirected graph $\mathcal{G}$, denoted by $A_{\mathcal{G}} = [a_{ij}]$, is an $n \times n$ symmetric matrix defined entry-wise as $a_{ij} = 1$ if nodes v_i and v_j are adjacent and $a_{ij} = 0$ otherwise.[1] The powers of the adjacency matrix is related to walks in a graph. In particular we have the following results [18]:

Lemma 2.1: The number of closed walks of length α joining node v_i to itself is given by the i-th diagonal entry of the matrix $A_{\mathcal{G}}^{\alpha}$.

Corollary 2.2: Let $\mathcal{G}$ be a simple graph. Denote by T_i and Q_i the number of triangles and quadrangles touching node v_i, respectively. Then $(A_{\mathcal{G}})_{ii} = 0$, $(A_{\mathcal{G}}^2)_{ii} = \deg v_i$, $(A_{\mathcal{G}}^3)_{ii} = 2T_i$ and $\left(A_{\mathcal{G}}^4\right)_{ii} = 2Q_i + (\deg v_i)^2 + \sum_{j \in N_i} (\deg v_j - 1)$.

Arranging the node degrees on a diagonal matrix yields the degree matrix $D_{\mathcal{G}} = \operatorname{diag}(\deg v_i)$. Then, the *Laplacian matrix* $L_{\mathcal{G}}$ of a graph $\mathcal{G}$ can be defined by $L_{\mathcal{G}} = D_{\mathcal{G}} - A_{\mathcal{G}}$. Let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ be the eigenvalues of $L_{\mathcal{G}}$, where $\mathbf{1}$ is the vector of all ones. One can prove that $L_{\mathcal{G}}$ is positive semidefinite and $\lambda_1 = 0$. Furthermore, $\mathcal{G}$ is connected if and only if $\lambda_2 > 0$, or equivalently, if $\ker L_{\mathcal{G}} = \text{span}\{\mathbf{1}\}$ [18]. As a result, we have the following well-known result:

Theorem 2.3 ([19]): Consider a fixed undirected graph $\mathcal{G}$ on n nodes and let $\theta_i(t) \in \mathbb{R}$ denote the state variable of node i. Let $\theta(t) = [\theta_i(t)] \in \mathbb{R}^n$ be the vector of all states and assume $\dot{\theta}(t) = -L_{\mathcal{G}}\theta(t)$. Then the network $\mathcal{G}$ is connected if and only if,

$$\lim_{t \to \infty} \theta(t) = \frac{1}{n} \sum_{i=1}^{n} \theta_i(0)\mathbf{1} \in \text{span}\{\mathbf{1}\}. \tag{1}$$

[1]For simple graphs with no self-loops, $a_{ii} = 0$ for all i.

for all initial conditions $\theta(0) \in \mathbb{R}^n$.

Theorem 2.3 says that the graph $\mathcal{G}$ is connected if and only if all nodes eventually reach a consensus on their state values $\theta_i(t)$, for all initial conditions. Therefore, connectivity of a network $\mathcal{G}$ can be verified almost surely by comparing the asymptotic state values (1) of all agents, for any random initialization. Note that a similar result can be obtained by application of a *finite-time* maximum (or minimum) consensus [20].

A. Problem Definition

Consider a discrete-time sequence of graphs $\{\mathcal{G}(s)\}_{s \geq 1}$ where $s \in \{1, 2, \dots\}$ is the discrete time index. We denote by $\{\lambda_i(s)\}_{s \geq 1}$ the set of Laplacian eigenvalues of $\mathcal{G}(s)$. We define the k-th spectral moment of the Laplacian matrix of $\mathcal{G}(s)$ at time s as:

$$m_k(s) \triangleq \frac{1}{n} \sum_{i=1}^{n} \lambda_i^k(s).$$

Similarly, the k-th centralized spectral moment of the Laplacian can written as:

$$\begin{aligned} \bar{m}_k(s) &\triangleq \frac{1}{n} \sum_{i=1}^{n} \left(\lambda_i(s) - m_1(s)\right)^k \\ &= \sum_{r=0}^{k} \binom{k}{r} (-1)^{k-r} m_r(s) m_1^{k-r}(s). \end{aligned} \tag{2}$$

The first four centralized spectral moments of the Laplacian corresponds to the mean, variance, skewness and kurtosis of the eigenvalue spectrum and they play a central role in this paper. Define further the error function:

$$\text{CME}(\mathcal{G}(s)) = \sum_{k=0}^{4} \left[\left(\bar{m}_k(s)\right)^{1/k} - \left(\bar{m}_k^*\right)^{1/k} \right]^2, \tag{3}$$

where $\bar{m}_k^*$ denotes a given set of desired centralized moments. Since the k-th moment is the k-th power-sum of the Laplacian eigenvalues, we include the exponents $1/k$ in the above error function with the purpose of assigning the same dimension to the summands in (3). Then, the problem addressed in this paper is:

Problem 1: Given an initially connected graph $\mathcal{G}(0)$, design a distributed algorithm that iteratively adds or deletes links from $\mathcal{G}(s)$, so that the connectivity of $\mathcal{G}(s)$ is maintained for all time s and the error function CME($\mathcal{G}(s)$) is locally minimized for large enough s.

In what follows, we first propose a distributed algorithm to efficiently compute and update CME($\mathcal{G}(s)$) without any explicit eigendecomposition (Section III). Then, in Section IV, we propose a greedy algorithm where the most beneficial edge addition/deletion is determined based on a distributed agreement over all possible actions that satisfy network connectivity (Theorem 2.3). In this framework, the time variable s increases by one whenever an action is taken (i.e., an addition or deletion of a link). For simplicity, we assume that actions are taken one at a time, although this assumption can be relaxed to accommodate more complex action schemes.

III. Distributed Computation of Spectral Moments

In what follows, we assume that the agents in the network have very limited knowledge of the network topology. In particular, we assume that every agent v only knows the topology of the second-order neighborhood subgraph around it, $\mathcal{G}_2^v$. (This is the case, for example, for many online social networks, where each individual can retrieve a list of his friends' friends.) Then, computing the first four Laplacian spectral moments relies on counting the presence of certain subgraphs, such a triangles and quadrangles, in every agent's neighborhood and averaging these quantities via distributed average consensus. In particular, since the matrix trace operator is conserved under diagonalization (in general, under any similarity transformation) the first three spectral moments of the Laplacian matrix of a graph can be written as

$$\begin{aligned} m_k\left(L_{\mathcal{G}}\right) &= \frac{1}{N}\,\mathrm{tr}\,L_{\mathcal{G}}^k = \frac{1}{n}\,\mathrm{tr}\;\;\left(D_{\mathcal{G}}-A_{\mathcal{G}}\right)^k \\ &= \;\frac{1}{n}\sum_{p=0}^{k}\binom{k}{p}(-1)^p\;\mathrm{tr}\;(A_{\mathcal{G}}^p D_{\mathcal{G}}^{k-p}), \end{aligned} \tag{4}$$

for $k \leq 3$, where we have used the fact that the trace is preserved under cyclic permutations (i.e., tr ADD=tr DAD= tr DDA). We cannot use Newton's binomial expansion for the forth moment; nevertheless, we may still obtain the following closed form solution:

$$\begin{aligned} m_4\left(L_{\mathcal{G}}\right) &= \frac{1}{n}\mathrm{tr}\left(D_{\mathcal{G}}-A_{\mathcal{G}}\right)^4 \\ &= \;\frac{1}{n}\left[\mathrm{tr}\left(D_{\mathcal{G}}^4\right)-4\mathrm{tr}\left(D_{\mathcal{G}}^3A_{\mathcal{G}}\right)+4\mathrm{tr}\left(D_{\mathcal{G}}^2A_{\mathcal{G}}^2\right)\right. \\ &\quad \left.+2\mathrm{tr}\left(D_{\mathcal{G}}A_{\mathcal{G}}D_{\mathcal{G}}A_{\mathcal{G}}\right)-4\mathrm{tr}\left(D_{\mathcal{G}}A_{\mathcal{G}}^3\right)+\mathrm{tr}\left(A_{\mathcal{G}}^4\right)\right]. \end{aligned} \tag{5}$$

Expanding the traces that appear in (4) and (5) we get

$$\mathrm{tr}\left(D_{\mathcal{G}}^qA_{\mathcal{G}}^p\right)=\sum_{i=1}^{n}\left(\deg v_i\right)^q\left(A_{\mathcal{G}}^p\right)_{ii}$$

and

$$\begin{aligned} \mathrm{tr}\left(D_{\mathcal{G}}A_{\mathcal{G}}D_{\mathcal{G}}A_{\mathcal{G}}\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\deg v_ia_{ij}\right)\left(\deg v_ja_{ji}\right) \\ &= \;\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\deg v_i\deg v_j\right)a_{ij}=\sum_{i=1}^{n}\deg v_i\sum_{j\in\mathcal{N}_i}\deg v_j, \end{aligned}$$

which substituted back in equations (4) and (5) give the following expression for $k \leq 3$

$$m_k\left(L_{\mathcal{G}}\right)=\frac{1}{n}\sum_{i=1}^{n}\sum_{r=0}^{k}\binom{k}{r}(-1)^r\left(\deg v_i\right)^{k-r}\left(A_{\mathcal{G}}^r\right)_{ii}. \tag{6}$$

For $k = 4$, we can also simplify the Laplacian spectral moment, which now becomes

$$\begin{aligned} m_4\left(L_{\mathcal{G}}\right) &= \frac{1}{n}\sum_{i=1}^{n}\Big[\left(\deg v_i\right)^4-4\left(\deg v_i\right)^3\left(A_{\mathcal{G}}\right)_{ii}+ \\ &\quad +4\left(\deg v_i\right)^2\left(A_{\mathcal{G}}\right)_{ii}^2+2\deg v_i\sum_{j\in\mathcal{N}_i}\deg v_j- \\ &\quad -4\left(\deg v_i\right)\left(A_{\mathcal{G}}\right)_{ii}^3+\left(A_{\mathcal{G}}\right)_{ii}^4\Big]. \end{aligned} \tag{7}$$

Substituting the expressions for $\left(A_{\mathcal{G}}\right)_{ii}^r$ form Lemma 2.1 and Corollary 2.2 in equations (6) and (7) we obtain the first four spectral moments of the Laplacian matrix $L_{\mathcal{G}}$ as a function of the second-order neighborhood subgraphs only

$$m_1\left(L_{\mathcal{G}}\right)=\frac{1}{n}\sum_{i=1}^{n}\deg v_i, \tag{8a}$$

$$m_2\left(L_{\mathcal{G}}\right)=\frac{1}{n}\sum_{i=1}^{n}\left[\left(\deg v_i\right)^2+\deg v_i\right], \tag{8b}$$

$$m_3\left(L_{\mathcal{G}}\right)=\frac{1}{n}\sum_{i=1}^{n}\left[\left(\deg v_i\right)^3+3\left(\deg v_i\right)^2-2T_i\right], \tag{8c}$$

$$\begin{aligned} m_4\left(L_{\mathcal{G}}\right) &= \frac{1}{n}\sum_{i=1}^{n}\Big[\left(\deg v_i\right)^4+4\left(\deg v_i\right)^3+\left(\deg v_i\right)^2- \\ &\quad -\deg v_i+\left(2\deg v_i+1\right)\sum_{j\in N_i}\deg v_j-8T_i\deg v_i+2Q_i\Big]. \end{aligned} \tag{8d}$$

Note that the expressions for the spectral moments in equations (8) are all *averages* of locally measurable quantities (within a 2-hop neighborhood), namely, node degrees, triangles and quadrangles touching the node. Hence, we can apply consensus and use the result of Theorem 2.3 to obtain the first four moments in a distributed way.

IV. Distributed Control of Spectral Moments

A. Possible Local Actions

The possible actions (or control variables) we consider are local link *additions* and local link *deletions*. A link addition is *local* if it connects a node with another node within its second-order neighborhood. Since agents in the network only know their local neighborhood, a fully distributed algorithm must limit edge additions to be local. (One could extend the algorithm to allow connections between nodes being further than two hops away, but this option would require much more computation and communication.)

Let $\mathcal{N}_1^i(s)$ and $\mathcal{N}_2^i(s)$ denote the sets of neighbors and two-hop neighbors of node i at time $s \geq 0$, respectively. Since any of the two nodes adjacent to a link can take an action to delete that link, we need to decide which of the two nodes has the authority to delete the link. To avoid ambiguities, we define the set of edges that node i has authority to remove as: $\mathcal{E}_d^i(s) \triangleq \left\{(i,j)\in\mathcal{E}(s)\mid j\in\mathcal{N}_1^i(s),\ i>j\right\}$. Similarly, to disambiguate between nodes adding a (still non-existing) link between them, we define the set of potential edges that node i can create as: $\mathcal{E}_a^i(s) \triangleq \left\{(i,j)\in\mathcal{E}(s)\mid j\in\mathcal{N}_2^i(s)\backslash\mathcal{N}_1^i(s),\ i>j\right\}$[2]. In other words, node i can either *add* a link $(i,j)\in\mathcal{E}_a^i(s)$, or *delete* a link $(i,j)\in\mathcal{E}_d^i(s)$. Note that link deletions may violate network connectivity. In this case, those link deletions should be excluded from the set of allowable actions. In the next two sections we address the cases of link deletions and link additions separately.

[2]Note that since the indices of all nodes in the network are distinct, this definition results in a unique assignment of links to nodes.

1) Link Deletions: Network connectivity is typically inferred from the number of trivial eigenvalues of the Laplacian matrix. However, such approaches are not applicable in our framework, since we assume no global knowledge of the network topology $\mathcal{G}(s)$, but only knowledge of local neighborhoods. Instead, we employ finite-time-maximum consensus which is a distributed algorithm and converges to equal values on nodes belonging to the same connected component of a graph (Theorem 2.3). Therefore, if deletion of a link violates connectivity, both nodes adjacent to that link will lie in different connected components and will have different consensus values. In what follows, we extend this idea to simultaneously checking connectivity for all possible edge deletions in the graph using a single high-dimensional consensus algorithm.

Consider node j that has authority to remove any of the links in the set $\mathcal{E}_d^j(s)$. Each one of these links needs to be checked with respect to connectivity and each connectivity verification relies on a scalar consensus update, according to Theorem 2.3. Therefore, checking all links in $\mathcal{E}_d^j(s)$ requires $|\mathcal{E}_d^j(s)|$ consensus updates.[3] We associate with every link in $\mathcal{E}_d^j(s)$ a consensus variable, and stacking all these variables in a vector we obtain the state vector $\mathbf{x}_{jj}(s) \in \mathbb{R}^{|\mathcal{E}_d^j(s)|}$. Running a distributed consensus over the network, requires participation of all other nodes $i \neq j$. This is possible by defining the state variables $\mathbf{x}_{ij}(s) \in \mathbb{R}^{|\mathcal{E}_d^j(s)|}$. All vectors $\mathbf{x}_{ij}(s)$ are initialized randomly and are updated by node i according to the following maximum consensus:
Case I: If $(i,j) \notin \mathcal{E}_d^i(s) \cup \mathcal{E}_d^j(s)$, i.e., if nodes i and j are not neighbors, then

$$\mathbf{x}_{ij}(s) := \max_{k \in \mathcal{N}_1^i(s)} \{\mathbf{x}_{ij}(s), \mathbf{x}_{kj}(s)\}, \tag{9}$$

with the maximum taken elementwise on the vectors,
Case II: If $(i,j) \in \mathcal{E}_d^j(s)$, i.e., if nodes i and j are neighbors and node j has authority over link (i,j), then

$$[\mathbf{x}_{ij}(s)]_{(i,j)} := \max_{k \in \mathcal{N}_1^i(s) \setminus \{j\}} \left\{[\mathbf{x}_{ij}(s)]_{(i,j)}, [\mathbf{x}_{kj}(s)]_{(i,j)}\right\}, \tag{10}$$

and

$$[\mathbf{x}_{ij}(s)]_{(l,j)} := \max_{k \in \mathcal{N}_1^i(s)} \left\{[\mathbf{x}_{ij}(s)]_{(l,j)}, [\mathbf{x}_{kj}(s)]_{(l,j)}\right\}, \tag{11}$$

for $l \neq i$, where $[\mathbf{x}_{kj}(s)]_{(l,j)} \in \mathbb{R}$ denotes the entry of $\mathbf{x}_{kj}(s)$ corresponding to the link (l,j),
Case III: If $(i,j) \in \mathcal{E}_d^i(s)$, i.e., if nodes i and j are neighbors and node i has authority over link (i,j), then

$$[\mathbf{x}_{ii}(s)]_{(i,j)} := \max_{k \in \mathcal{N}_1^i(s) \setminus \{j\}} \left\{[\mathbf{x}_{ii}(s)]_{(i,j)}, [\mathbf{x}_{ki}(s)]_{(i,j)}\right\}. \tag{12}$$

Once consensus (9)–(12) has converged, node i compares the entries $[\mathbf{x}_{ii}(s)]_{(i,j)}$ and $[\mathbf{x}_{ji}(s)]_{(i,j)}$ for all $(i,j) \in \mathcal{E}_d^i(s)$. Since, violation of connectivity due to deletion of the link (i,j) would result in nodes i and j being in different connected components of the network, $[\mathbf{x}_{ii}(s)]_{(i,j)} = [\mathbf{x}_{ji}(s)]_{(i,j)}$ implies that the reduced network is still connected. Hence, we can define a set

$$\mathcal{E}_\sigma^i(s) \triangleq \left\{(i,j) \in \mathcal{E}_d^i(s) | [\mathbf{x}_{ii}(s)]_{(i,j)} = [\mathbf{x}_{ji}(s)]_{(i,j)}\right\}, \tag{13}$$

containing the *safe* links adjacent to node i that if deleted, connectivity is maintained.

[3]We define by $|X|$ the cardinality of the set X.

Algorithm 1 Connectivity verification

Require: $\mathbf{x}_{ij} \in \mathbb{R}^{|\mathcal{E}_d^i|}$ for all $j \in \mathcal{V}$;
Require: $T_{ij} = [0 \dots 1_i \dots 0]^T$, $\forall\, j \in \mathcal{V}$;
1: **if** $\exists\, j \in \mathcal{V}$ such that $\min\{T_{ij}\} = 0$ **then**
2: Update $\mathbf{x}_{ij}$ by (9)–(12);
3: $T_{ij} := \max_{k \in \mathcal{N}_i^1}\{T_{ij}, T_{kj}\}$;
4: **else if** $\min\{T_{ij}\} = 1$ for all $j \in \mathcal{V}$ **then**
5: Update $\mathcal{E}_\sigma^i$ by (13);
6: **end if**

2) Connectivity Verification: The connectivity verification of link deletions, discussed in Section IV-A.1, is illustrated in Alg. 1. Convergence of the finite-time consensus (9)–(12) is captured by a vector of tokens $T_{ij}(s) \in \{0,1\}^n$, initialized as $T_{ij}(s) := [0 \dots 1_i \dots 0]^T$ for all $j \in \mathcal{V}$ and indicating that node i has initialized the consensus variables for link deletions for which node j is responsible. In particular, when all tokens of all nodes have been collected (line 4, Alg. 1), then consensus has converged and the set of safe link deletions $\mathcal{E}_\sigma^i(s)$ can be computed (line 5, Alg. 1). Note that node i does not need to know the neighbor sets of its non-neighbors in the network, neither their size. Instead, the vectors $\mathbf{x}_{ij}(s)$ are initialized both in values and dimension as soon as vectors $\mathbf{x}_{kj}(s)$ are received from a neighbor $k \in \mathcal{N}_1^i(s)$. Clearly, $\mathbf{x}_{jj}(s)$ are initialized first and then propagated in the network via maximum consensus until the information they contain reaches node i.

B. Most Beneficial Local Action

As discussed in Problem 1, the objective of this work is to minimize the error function CME($\mathcal{G}(s)$). For this we propose a greedy algorithm, which for every time s selects the action that maximizes the quantity CME($\mathcal{G}(s)$) − CME($\mathcal{G}(s+1)$), if such an action exists, and terminates if no such action exists. By construction, such an algorithm converges to a network that *locally* minimizes CME($\mathcal{G}(s)$), while in Section V, we show that it performs well in practice too.

In what follows we first compute the effect of a link addition or deletion on the four first spectral moments. Although distributed consensus could be used to compute the new moments resulting from each possible action, as in Section III, this would clearly be computationally very expensive. Instead, we can achieve this goal locally and with minor computational overhead, based on the observation that the addition or removal of an edge (i,j) only influences the degrees of nodes i and j, as well as and the triangles and quadrangles touching their neighboring nodes. Hence, agents i and j can communicate to compute the net increment in the spectral moments due to the addition or deletion of edge

(i,j). In particular, we get the following expressions for the increments in the first three moments

$$\Delta m_1^{\pm(i,j)} = \pm\frac{2}{n},$$
$$\Delta m_2^{\pm(i,j)} = \frac{2}{n}\left[1 \pm (d_i + d_j + 1)\right],$$
$$\Delta m_3^{\pm(i,j)} = \frac{1}{n}\left[(3 \pm 6)(d_i + d_j) \pm 3(d_i^2 + d_j^2) + (6 \pm 2) \mp 6T_{ij}\right]$$

where the notation $\pm(i,j)$ indicates a link addition $(+)$ or deletion $(-)$ and the dependence on time s has been dropped for simplicity. (Similarly, one can obtain a complicated closed-form expression for $\Delta m_4^{\pm(i,j)}$, which we omit due to space limitations.) Then, agent's i copy of the k-th spectral moment $m_k^i(s)$ becomes

$$m_k^{\pm(i,j)}(s) = m_k^i(s) + \Delta m_k^{\pm(i,j)}(s),$$

and the associated centralized moment $\bar{m}_k^{\pm(i,j)}(s)$ can be computed using (2). Then, for all possible actions discussed in Section IV-A, agent i computes the error function

$$\text{CME}_{\pm(i,j)}(s) = \sum_{k=0}^{4}\left[\left(\bar{m}_k^{\pm(i,j)}(s)\right)^{1/k} - (\bar{m}_k^*)^{1/k}\right]^2.$$

Then, the local *most beneficial action* to the target centralized moments, namely, the action that results in the maximum decrease in the error function $\text{CME}(\cdot)$, can be defined by[4]

$$\nu_i(s) \triangleq \max\left\{\underset{j}{\text{argmin}}\ \left(\text{CME}_{\pm(i,j)}(s) - \text{CME}_i(s)\right)\right\},$$

where $\text{CME}_i(s)$ denotes agent i's copy of $\text{CME}(s)$, and the largest decrease in the error associated with action $\nu_i(s)$ becomes:

$$\text{CME}_i(s) = \text{CME}_{\pm(i,\nu_i(s))}(s)$$

if $\min_j(\text{CME}_{\pm(i,j)}(s) - \text{CME}(s)) \leq 0$ and $\text{CME}_i(s) = D$, otherwise. Note that $\text{CME}_i(s)$ is nontrivially defined only if the exists a link adjacent to node i that if added or deleted decreases the error function $\text{CME}(\cdot)$. Otherwise, a large value $D > 0$ is assigned to $\text{CME}_i(s)$ to indicate that this action is not beneficial to the final objective. We can include all information of a best local action in the vector

$$\mathbf{v}_i(s) \triangleq \left[i\ \nu_i(s)\ \text{CME}_i(s)\ \bar{\mathbf{m}}_{\pm(i,\nu_i(s))}(s)\right]^T \in \mathbb{R}^7$$

containing the local best action $(i,\nu_i(s))$, the associated distance to the desired moments $\text{CME}_i(s)$, and the vector of centralized moments $\bar{\mathbf{m}}_{\pm(i,\nu_i(s))}(s)$ due to this action. In the following section we discuss how to compare all local actions $\mathbf{v}_i(s)$ for all nodes $i \in \mathcal{V}$ to obtain the one that decreases the distance to the desired moments the most.

[4]The max in the expression bellow indicates that in case of ties in the min, the highest index node wins.

Algorithm 2 Globally most beneficial action

Require: $\mathbf{v}_i \triangleq [i\ \nu_i\ \text{CME}_i\ \bar{\mathbf{m}}_{\pm(i,\nu_i)}]^T$;
Require: $T_i := [0 \dots 1_i \dots 0]^T$;
1: **if** $\min\{T_i\} = 0$ **then**
2: $\quad\mathbf{v}_i := \mathbf{v}_j$, with $j = \max\{\text{argmin}_{k\in\mathcal{N}_1^i}\{[\mathbf{v}_i]_3, [\mathbf{v}_k]_3\}$;
3: $\quad T_i := \max_{j\in\mathcal{N}_1^i}\{T_i, T_j\}$;
4: **else if** $\min\{T_i\} = 1$ and $[\mathbf{v}_i]_3 < D$ **then**
5: $\quad$Update $\mathcal{N}_i$, $\bar{\mathbf{m}}_i$ and CME_i according to (14)–(17);
6: **else if** $\min\{T_i\} = 1$ and $[\mathbf{v}_i]_3 = D$ **then**
7: $\quad$No beneficial action. Algorithm has converged;
8: **end if**

C. From Local to Global Action

To obtain the overall most beneficial action, all local actions need to be propagated in the network and compared against each other. For this we require minimal storage and communication as well as no propagation of any information regarding the network topology. As in Section IV-A, we achieve this goal using a finite time minimum consensus algorithm.

In particular, the desired local actions $\mathbf{v}_i(s)$ are propagated in the network, along with vectors of tokens $T_i(s) \in \{0,1\}^n$, initialized as $T_i(s) := [0 \dots 1_i \dots 0]^T$, indicating that node i has transmitted its desired action. During every iteration of the algorithm, every node i communicates with its neighbors and updates its vector of tokens $T_i(s)$ (line 3, Alg. 2), as well as its desired action $\mathbf{v}_i(s)$ with the action $\mathbf{v}_j(s)$ corresponding to the node j that contains the smallest distance to the target moments $[\mathbf{v}_j(s)]_3$, i.e.,

$$j \in \text{argmin}_{k\in\mathcal{N}_1 i(s)}\{[\mathbf{v}_i(s)]_3, [\mathbf{v}_k(s)]_3\}.$$

In case of ties in the distances to the targets $[\mathbf{v}_j(s)]_3$, i.e., if $\text{argmin}_{k\in\mathcal{N}_1^i(s)}\{[\mathbf{v}_i(s)]_3, [\mathbf{v}_k(s)]_3\}$ contains more than one nodes, then the node j with the largest label is selected (line 2, Alg. 2). Note that line 2 of Alg. 2 is essentially a minimum consensus update on the entries $[\mathbf{v}_i(s)]_3$ and will converge to a common outcome for all nodes when they have all been compared to each other, which is captured by the condition $\min_{j=1}^n T_{ij}(s) = 1$ (lines 4 and 6, Alg. 2). When the consensus has converged, if there exists a node whose desired action decreases the distance to the target moments, i.e., if $[\mathbf{v}_i(s)]_3 < D$ (line 4, Alg. 2), then Alg. 2 terminates with a greedy action and node i updates its set of neighbors $\mathcal{N}_1^i(s)$ and vector of centralized moments $\bar{\mathbf{m}}_i(s)$ (line 5, Alg. 2). If the optimal action is a link addition, i.e., if $[\mathbf{v}_i(s)]_2 \notin \mathcal{N}_1^i(s)$, then

$$\mathcal{N}_1^i(s+1) := \mathcal{N}_1^i(s) \cup \{[\mathbf{v}_i(s)]_2\}. \tag{14}$$

On the other hand, if the optimal action is a link deletion, i.e., if $[\mathbf{v}_i(s)]_2 \in \mathcal{N}_1^i(s)$, then

$$\mathcal{N}_1^i(s+1) := \mathcal{N}_1^i(s) \setminus \{[\mathbf{v}_i(s)]_2\}. \tag{15}$$

In all cases, the centralized moments and error function are updated by

$$\bar{\mathbf{m}}_i(s+1) := [\mathbf{v}_i(s)]_{4:7} \tag{16}$$

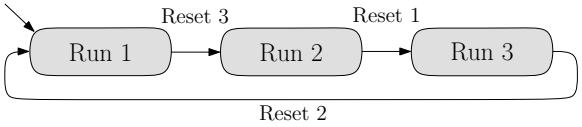

Fig. 1. Synchronization: Assume node i is in run 1. Necessary for node i to transition to run 2 is that all other nodes are also in run 1, since otherwise node i will be missing tokens from the nodes that are not in run 1 yet (currently in run 3) and Alg. 2 will not be able to converge. Once node i transitions to run 2, it initializes all variables for that run with the latest values from run 1, while it maintains the variables of run 1 for nodes that are still in run 1 and it clears all variables of run 3 since, no node is in this run any more.

and

$$\text{CME}_i(s+1) := [\mathbf{v}_i(s)]_3, \tag{17}$$

respectively, where $[\mathbf{v}_i(s)]_{4:7} = [[\mathbf{v}_i(s)]_4 \dots [\mathbf{v}_i(s)]_7]^T$. Finally, if all local desired actions increase the distance to the target moments, i.e., if $[\mathbf{v}_i(s)]_3 = D$ (line 6, Alg. 2), then no action is taken and the algorithm terminates with a network topology with almost the desired spectral properties. This is because no action exists that can further decrease the distance to the target moments.

D. Synchronization

Communication time delays, packet losses, and the asymmetric network structure, may result in runs of the algorithm starting asynchronously, outdated information being used for future decisions, and consequently, nodes reaching different decisions for the same run. In the absence of a common global clock, the desired synchronization is ideally *event triggered*, where by a triggering event we understand the time instant that a message $\text{Msg}[i]$ has been received by any of node i's neighbors $j \in \mathcal{N}_1^i$. We achieve such a synchronization by labeling every algorithm run in the set $\{1, 2, 3\}$ and requiring that all information exchange takes place among neighbors that are in equally labeled runs [21]. Essentially, "fast" nodes wait for their "slower" peers and, hence, all nodes are always synchronized in the sequence $\{1, 2, 3, 1, 2, 3, \dots\}$ (Fig. 1).

V. Numerical Simulations

In this section we illustrate our algorithm with several numerical examples.

Example 1 (Star networks): Consider a star network on 10 nodes. The first four central moments of the associated Laplacian matrix are: $\bar{m}_1^* = 1.8$, $\bar{m}_2^* = 7.56$, $\bar{m}_3^* = 54.14$, and $\bar{m}_4^* = 453.49$. Our objective is to control the topology of a randomly initialized network on 10 nodes so that it eventually has the same set of moments as the given star network. We observe in Fig. 2 that our algorithm decreases the error function (blue line) to zero in finite time. Similar performances are observed when we repeat this procedure for star networks of any size. Furthermore, although we are controlling the first four moments solely, the resulting network structures are exactly the star topologies whose moments we were trying to approximate. The perfect reconstruction observed in this case is due to the fact that a star graph is a extreme case in which the graph topology is uniquely defined by their eigenvalue spectrum. Moreover, if each agent in a star network has access to its second-order neighborhood, it has access to the complete star topology.

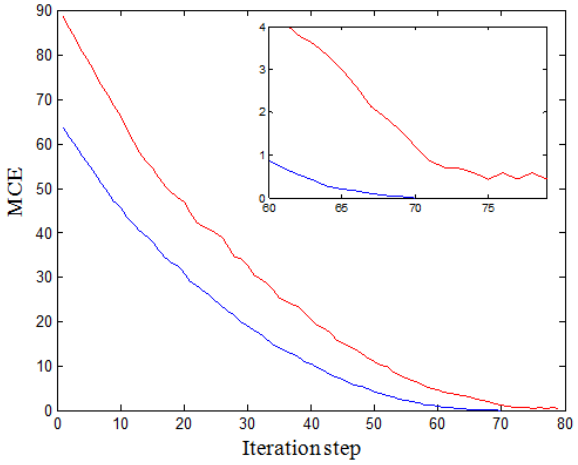

Fig. 2. Convergence of the error function $\text{CME}(\cdot)$ for the star graph (blue plot) and the two-stars graph (red plot). The subgraph in the upper right corner shows the behavior of the error function in a neighborhood of zero. Observe that our algorithm can match the first four moments of the star network with zero error in finite time, but can not exactly match the moments of a the two-stars graph, although the final error is very small.

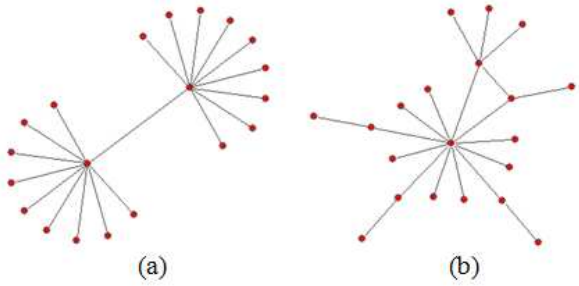

Fig. 3. Structures of the two-stars network (a) and the network returned by our algorithm (b).

Example 2 (Two-stars network): Although our approach works very well for star networks, the case of two-stars networks points out one of its weaknesses, namely, its limitation in modeling network communities. In this example, we consider two star graphs on 10 nodes each, and connect their two central hubs with a link. The resulting graph is the two-stars graph shown in Fig. 3(a). As before, we initialize our algorithm with a random graph on 20 nodes and try to approximate the first four central moments of the two-stars graph. In Fig. 2, we observe that the error function (red line) quickly reaches a neighborhood of zero but does not reach zero exactly. Therefore, although our algorithm tries to generate the two hubs in the two-star network, its local nature will not allow it recover the highly-structured two-stars graph. Instead, it returns the final network shown in Fig. 3(b). Nevertheless, the eigenvalue spectra between the desired two-star network and the network in Fig. 3(b) are still very similar, as shown in Fig. 4.

Example 3 (Chain vs. ring networks): The objective of

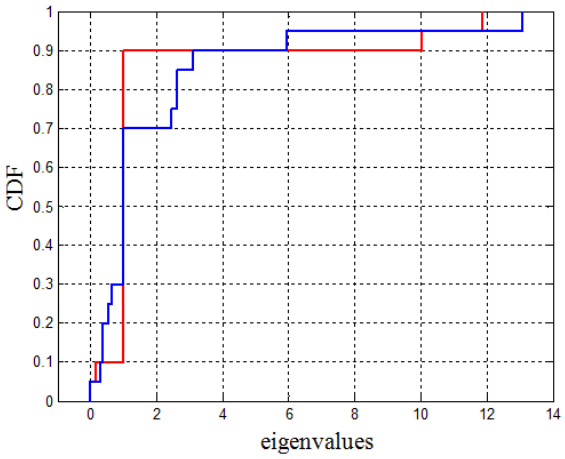

Fig. 4. Empirical cumulative distribution functions for the eigenvalues of the two-stars graph (blue) and the graph returned by our algorithm (red).

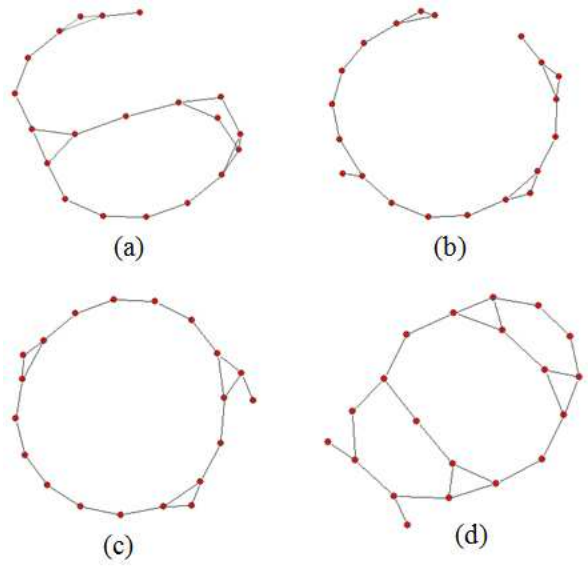

Fig. 5. Networks returned by our algorithm when trying to match the first four central moments of a ring on 20 nodes.

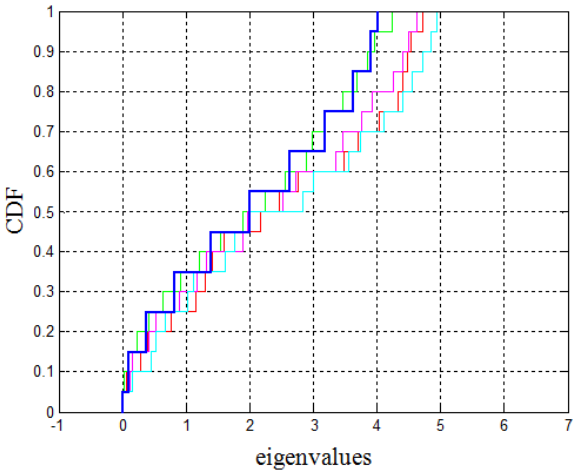

Fig. 6. Empirical cumulative distribution of eigenvalues for the ring graph with 20 nodes (blue plot) and the graphs in Fig. 5(a) (red), Fig. 5(b) (green), Fig. 5(c) (magenta) and Fig. 5(d) (cyan).

this example is to illustrate how two structurally very similar (but topologically different) target graphs, such as a chain and a ring, may affect the performance of our algorithm. In particular, if we run our algorithm to control the first four centralized moments of an initially random graph towards the moments of a chain graph, we observe how the error function converges exactly to zero in finite time. Furthermore, the final result of our algorithm is an exact reconstruction of the chain graph. Nevertheless, when transforming the target graph from a chain graph into a ring graph (by adding a single link), an exact reconstruction is very difficult. In Fig. 5, we illustrate some graphs returned by our algorithm for different initial conditions when we control the set of moments toward the moments of a ring network on 20 nodes. Observe that, although the algorithm tends to create long cycles and the majority of nodes have degree two, it fails to recreate the exact structure of the ring graph due to the local nature of the algorithm (as in Example 2). However, although the structure of the resulting networks is not exactly the desired ring graph, their spectral properties are remarkably close to those of a ring. In Fig. 6, we illustrate the empirical cumulative distribution functions of the eigenvalues of the ring graph (blue plot), versus the four empirical cumulative distribution functions corresponding to the graphs in Fig. 5.

Example 4 (Small-Worlds): In our final example, we use our algorithm to control the moments of a randomly generated graph to those of a small-world network. We consider small-world graphs as defined in [22], namely, we take a ring of n nodes, and connect each node in the ring with all the nodes in its 3-hop neighborhood. Then, we randomly rewire a fraction of the resulting edges with probability p as proposed by Watts and Strogatz [22]. Our objective is to approximate the first four centralized moments of a random instance of a small world graph with $n = 40$ nodes and link probability $p = 1/n$. We observed a fast convergence of the error to a neighborhood of zero, i.e., $\text{CME}(27 \text{ iterations}) = .0009$, which suggests (although does not guarantee) a good approximation between the spectra of the target small-world graph and the graph returned by our algorithm. We repeated this process for a link probability $p = 4/n$ and similar results were obtained. We should note, however, that although the spectral properties between the target small-world graphs and the graphs returned by our algorithm are remarkably similar (Fig. 7), we expect the structures to be quite different as in Example 3 (although this difference is not easy to observe from a direct visualization of both graph structures).

VI. Conclusions and Future Research

In this paper, we have described a fully decentralized algorithm that iteratively modifies the structure of a network of agents with the objective of controlling the spectral moments of the Laplacian matrix of the network. Although we assume that each agent has access to local information regarding the graph structure, we show that the group is able to collectively aggregate their local information to take a global optimal decision. This decision corresponds to the most beneficial

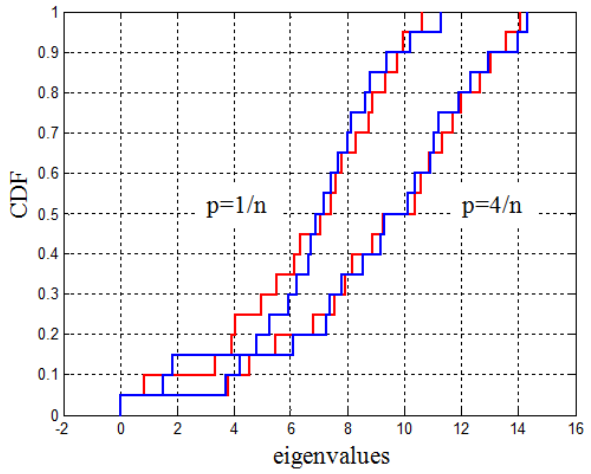

Fig. 7. Empirical cumulative distrubution of eigenvalues for the small-world graphs in the example for $p = 1/n$ (left) and $p = 4/n$ (right).

link addition/deletion in order to minimize an error function that involves the first four Laplacian spectral moments of the network. The aggregation of the local information is achieved via gossip algorithms, which are also used to ensure network connectivity throughout the evolution of the network.

Future work involves identifying sets of spectral moments that are reachable by our control algorithm. (We say that a sequence of spectral moments is reachable if there exists a graph whose moments match the sequence of moments.) Furthermore, we observed that fitting a set of low-order moments does not guarantee a good fit of the complete distribution of eigenvalues. In fact, there are important spectral parameters, such as the algebraic connectivity, that are not captured by a small set of spectral moments. Nevertheless, we observed in numerical simulations that fitting the first four moments of the eigenvalue spectrum often achieves a good reconstruction of the complete spectrum. Hence, a natural question is to describe the set of graphs most of whose spectral information is contained in a relatively small set of low-order moments.

REFERENCES

[1] N. Wiener, *The Mathematics of Self-Organising Systems. Recent Developments in Information and Decision Processes*, Macmillan, 1962.

[2] H. Haken, *Synergetics: An Introduction*, 3^{rd} Edition, Springer-Verlag, 1983.

[3] M.O. Jackson, *Social and Economic Networks*, Princeton University Press, 2008.

[4] V.M. Preciado, *Spectral Analysis for Stochastic Models of Large-Scale Complex Dynamical Networks*, Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., MIT, Cambridge, MA, 2008.

[5] D.J. Aldous, *Some Inequalities for Reversible Markov Chains*, J. London Math. Soc. (2)25, pp. 564-576, 1982.

[6] L.M. Pecora, and T.L. Carroll, "Master Stability Functions for Synchronized Coupled Systems," *Physics Review Letters*, vol. 80(10), pp. 2109-2112, 1998.

[7] V.M. Preciado, and G.C. Verghese, "Synchronization in Generalized Erdös-Rényi Networks of Nonlinear Oscillators," *Proc. of the 44th IEEE Conference on Decision and Control*, pp. 4628-4633, 2005.

[8] M. Draief, A. Ganesh, and L. Massoulié, "Thresholds for Virus Spread on Networks," *Annals of Applied Probability*, vol. 18, pp. 359-378, 2008.

[9] V.M. Preciado, and A. Jadbabaie, "Spectral Analysis of Virus Spreading in Random Geometric Networks," *Proc. of the 48th IEEE Conference on Decision and Control*, 2009. (to appear).

[10] N.A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, 1997.

[11] A. Fax, and R. M. Murray, "Information Flow and Cooperative Control of Vehicle Formations," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1465-1476, 2004.

[12] R. Olfati-Saber, and R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520-1533, 2004.

[13] A. Ghosh, and S. Boyd, "Growing Well-Connected Graphs," *Proc. of the 45th IEEE Conference on Decision and Control*, pp. 6605-6611, 2006.

[14] R. Grone, R. Merris, and V.S. Sunder, "The Laplacian Spectrum of a Graph," *SIAM Journal Matrix Analysis and Applications*, vol. 11, pp. 218-238, 1990.

[15] Y. Kim, and M. Mesbahi, "On Maximizing the Second-Smallest Eigenvalue of a State Dependent Graph Laplacian," *IEEE Transactions on Automatic Control*, vol. 51, pp. 116-120, 2006.

[16] M.C. DeGennaro, and A. Jadbabaie, "Decentralized Control of Connectivity for Multi-Agent Systems," *Proc. of the 45th IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006, pp. 3628-3633.

[17] D. Kempe, and F. McSherry, "A Decentralized Algorithm for Spectral Analysis," *Journal of Computer and System Science*, vol. 74, pp. 70-83, 2008.

[18] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, 2^{nd} Edition, 1993.

[19] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48(6), pp. 988-1001, 2003.

[20] J. Cortes, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44(3), pp. 726-737, 2008.

[21] M. M. Zavlanos, and G. J. Pappas, "Distributed Connectivity Control of Mobile Networks," *IEEE Transactions on Robotics*, vol. 24(6), pp. 1416-1428, 2008.

[22] D.J. Watts, and S. Strogatz, "Collective Dynamics of Small World Networks," *Nature*, vol. 393, pp. 440-42, 1998.